Программирование

Python

Кафедра ИВТ и ПМ
ЗабГУ

2022

План

- Синтаксис языка
- Арифметическое выражение
- Оператор присваивания
- Оператор вызова функции
- Условный оператор
- Пустой оператор
- Приоритет операторов

Синтаксис

Синтаксис языка программирования — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом.

Синтаксис

Пример синтаксического правила.

Для вызова функции с параметром x следует указать имя функции (её идентификатор), затем в круглых скобках выражение - пераметр. Допустимо ставить пробелы между идентификаторами и скобками.

```
sin ( x )
```

Синтаксические ошибки

В компиляруемых языках синтаксические ошибки выявлятся во время компиляции программы. Компилятор может обнаружить *все* синтаксичекий ошибки в программе.

В интерпретируемых языках - во время её выполнения. При обнаружении синтаксической ошибки программа аварийно завершает свою работу. Интерпретатор как правило может указать только на одну (первую) синтаксическую ошибку.

Если же какая-то часть программы не воплнилась, то она не была проверена на наличие синтаксических ошибок.

Синтаксические ошибки. Примеры

```
sin x
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

пропущен операнд после оператора сложения

```
x + 2 = 7
```

Синтаксические ошибки. Примеры

```
sin x
```

пропущены круглые скобки вокруг x

```
20 / 7 +
```

пропущен операнд после оператора сложения

```
x + 2 = 7
```

слева от оператора присваивания обязательно должна быть только переменная!

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
    20 / 7 +
             ^
SyntaxError: invalid syntax
```

Синтаксические ошибки

Интерпретатор Python указывает место и тип синтаксической ошибки.

```
File "program.py", line 1
    20 / 7 +
             ^
SyntaxError: invalid syntax
```

Синтаксичекая ошибка в файле *program.py* в первой строчке.

Особенности синтаксиса языка Python

- После инструкций точка с запятой не ставится
- Одна строка - одна инструкция
- Конец строки - конец инструкции
- Пустые строки интерпретатором игнорируются

Особенности синтаксиса языка Python

Пример последовательности инструкций (операторов):

```
a = 42
b = 13
c = a + b

print("a + b = ", c)
```

Отдельные логические части последовательностей операторов можно разделять пустыми строками.

Для улучшения читаемости знаки операций можно отделять пробелами.

Выражения

Выражение (expression) – комбинация значений, констант, переменных, операций и функций, которая может быть интерпретирована в соответствии с правилами конкретного языка.

Арифметическое выражение – выражение результат которого имеет числовой тип (целый или вещественный)

Выражения

Примеры выражений

```
2

3 + 1

x  # x - объявленная ранее переменная

22 + x / 2 - y**0.5

a and b  # a и b переменные логического типа

sin(pi)

input()
```

Оператор присваивания

Оператор присваивания (assignment statement)- оператор связывающий переменную некоторым значением.

Формат:

```
переменная = выражение
```

Во время выполнении *операции присваивания* сначала вычисляется результат выражения, а затем он связывается с переменной.

Оператор присваивания

Оператор присваивания обозначается знаком равенства, но это не то же самое, что равенство в математическом смысле.

Корректная с точки зрения языка программирования запись, но не не имеющее смысла математическое уравнение:

```
# для ранее определённой перменной x
x = x + 1
```

Корректное математическое уравнение, но не корректная запись с точки зрения языка программирования:

```
x + 1 = x
```

Оператор присваивания. Примеры

ПРАВИЛЬНО

```
x = 20
y = 20/8 - 6 + x

z = x

x = x + 1

y = sin( 0 )

# ничего не изменит
x = x
```

НЕПРАВИЛЬНО

```
20 = 7

x + 1 = 8

x + y = 9

sin(x) = 0
```

Оператор вызова функции ()

Функция — фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.

Формат

`идентификатор` (`параметры`)

идентификатор - идентификатор функции;
параметры - выражения разделённые запятой.

Параметры могут отсутствовать.

Оператор вызова функции ()

Примеры

```
input()  # нет параметров

print("Python!")  # один параметр

log(9, 3)  # два параметра
# = 2.0
```

Примеры функции

```
# функцию (как и переменную) нужно сперва объявить
def my_function():
    print("This is my very fist function!")
    print("2 x 2 = 5")

# а потом использовать - вызвать
my_function()

# на экран будет выведено
>>> This is my very fist function!
>>> 2 x 2 = 5
```

Оператор вызова функции ()

Если написать () послед обычной переменной или значения, Python будет думать, что вы хотите вызывать это как функцию.

```
x = 42
>>> TypeError: 'int' object is not callable
# нельзя вызывать переменную типа int как функцию

3.14()
>>> TypeError: 'float' object is not callable
```

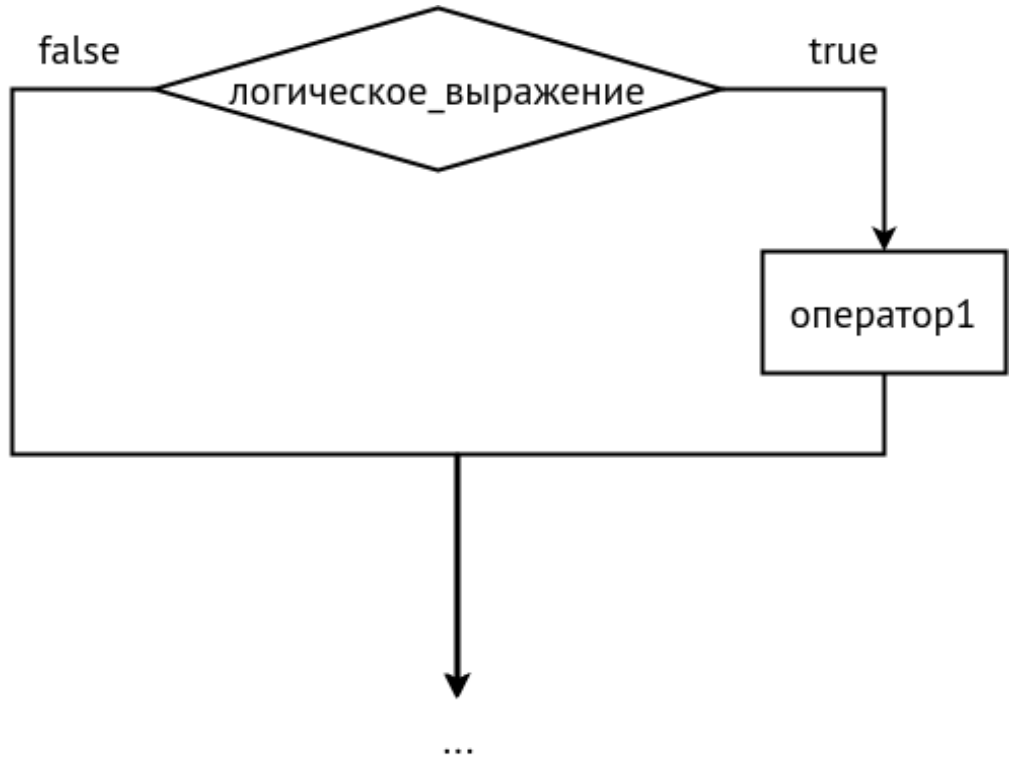
Оператор вызова функции ()

- Каждая функция имет своё количество параметров (аргументов).
- Некоторые функции могут принимать разное количество аргументов
  например: print(), print("Hello, World!") и print("Hello, "World!")
- Функция ождидает, что переданные ей параметры будут иметь конкретный тип
  Наприме параметров функции sin() должно быть число, а не строка
- Некоторые функции не имеют жестких требований к типы передаваемых им параметров

Условный оператор

Сокращённая форма

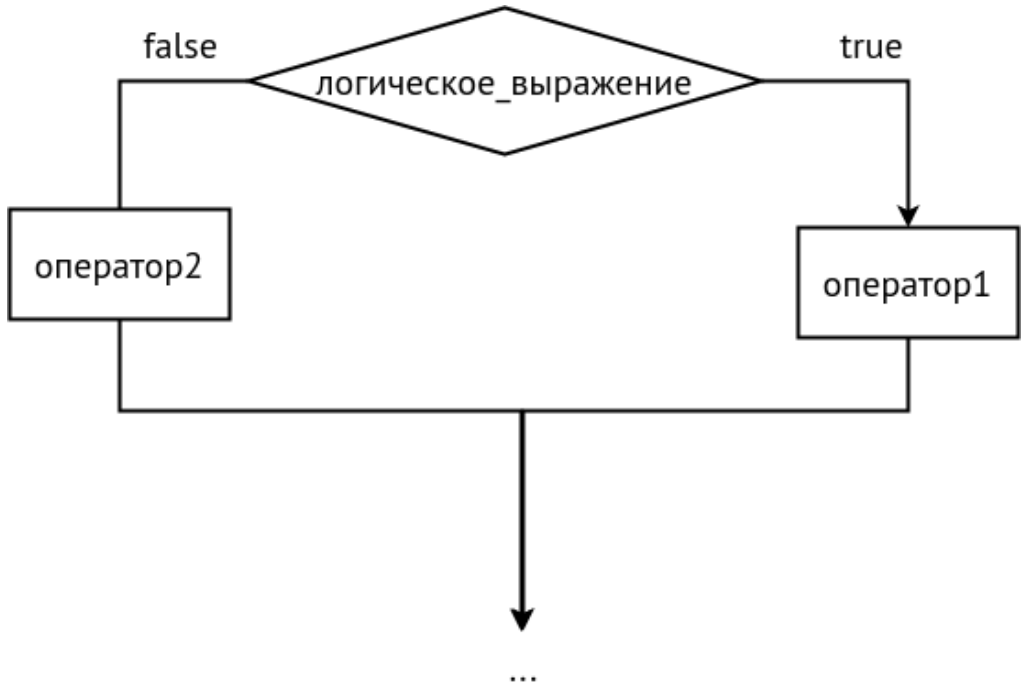
```
if логическое_выражение :
    оператор1
```



Условный оператор

Полная форма

```
if логическое_выражение :
    оператор1
else:
    оператор2
```



Условный оператор

Операторов может быть несколько.
Все они должны выделятся отступами

```
if лог.выражение:
    оператор1
    оператор2
    оператор3

оператор4
```

оператор4 не относится к блоку `if` и будет выполнен в любом случае.

Логические операции

Внутри оператора `if` условие записывется в виде логического выражения. Результат логического выражения имеет логический тип.

Таблицы истинности логических операций: not (НЕ), and (И), or (ИЛИ), xor (исключающее или)

NOT

x	x'
0	1
1	0

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Логические выражения

В Python переменная или выражение логического типа могут принмать только два значения: `True`, `False`

Операции с логическим типом в Python: `not`, `and`, `or`, `^` (xor)

```
x = 5
y = 6
# примеры логических выражений:
x == 5 and y == 6    # True
x == 5 and y == 20   # False
x == 50 and y == 6   # False

x >= 5               # True
x == 5 or x == 6     # True
x == 5 and x > 5     # False при любых значениях x
x >= 5               # True
```

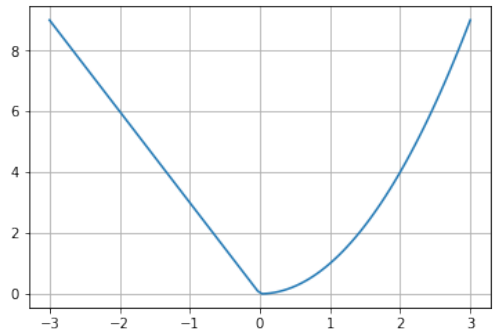
Логические выражения

```
x = 5
y = 6
z = 5

x < y and x == z   # True
x != y   # True
x == 6 or x == 7   # True
x < y < z   # False

True ^ False   # True
```

Условный оператор. Примеры

Вычислить математическое выражение.

$$y = \begin{cases} x^2, x > 0 \\ -2x, x \leq 0 \end{cases}$$




```
x = float( input("x = ") )
y = None

if x>0:
    y = x**2
else:
    y = -3*x

print("y = ", y)
```

Особенности синтаксиса языка Python

- Отступы - это часть синтаксиса языка
- Операторных скобок нет
- Уровни вложенности определяются отступами.
- Для одного уровня вложенности рекомендуется использовать отступ из 4-х пробелов.

```
if a > b:
    print("a > b")

while a > b:
    a = a + 1
```

Перед вложенными операторами всегда ставится двоеточие.

Условный оператор. Примеры

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

Условный оператор. Примеры

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

```
max = a
if b > max:
    max = b


print( "max = ", max )
```

Условный оператор. Примеры

Логические выражения в условном операторе. В зависимости от ответа пользователя вывести заполнить переменную случайным число или с клавиатуры.

```
from random import random

ans = input("Заполнить данные с помощью random? ")
x = None
if ans == "да" or ans == 'lf':  # lf - да в анг. раскладке
    x = random()*100 - 50
    print(x)
else:
    x = float( input("Введите вещественное число: ") )
```

Условный оператор. Примеры

Разделение на два условных оператора
Если варажение внутри условного оператора это логическое И

Один условный оператор

```
...
if x > 0 and y > 0:
  print("Числа положительны")
```

Два условных оператора, один вложен в другой

```
...
if x > 0:
  if y > 0:
    print("Числа положительны")
```

Условный оператор. Примеры

В какой честверти координатной плоскости находится точка с заданными координатами (x,y)?
Предположить что, координаты не могут быть нулями.

Условный оператор. Примеры

В какой честверти координатной плоскости находится точка с заданными координатами (x,y)?
Предположить что, координаты не могут быть нулями.

Сколько понадобится условных операторов?

Условный оператор. Примеры

В какой честверти координатной плоскости находится точка с заданными координатами (x,y)?
Предположить что, координаты не могут быть нулями.

Сколько понадобится условных операторов?

```
if x > 0:
    if y > 0:
        print("Первая четверть")
    else:  # y < 0
        print("Четвёртая четверть")
else: # x < 0
    if y > 0:
        print("Вторая четверть")
    else:  # y < 0
        print("Третья четверть")
```

Условный оператор. Примеры

В какой честверти координатной плоскости находится точка с заданными координатами (x,y)?
Предположить что, координаты не могут быть нулями.

Условный оператор. Примеры

В какой честверти координатной плоскости находится точка с заданными координатами (x,y)?
Предположить что, координаты не могут быть нулями.

Вариант решения задачи со сложным условием.

```
if x > 0 and y > 0:
    print("Первая четверть")
else:
    if x > 0 and y < 0:
        print("Четвёртая четверть")
    else:
        if x < 0 and y > 0:
            print("Вторая четверть")
        else:  # хм, остался только один вариант
               # значит эта ветка для x<0 and y<0
            print("Третья четверть")
```

Такой вариант кода немного труднее для чтения и понимания.

Условный оператор. Примеры

Определить находится ли точка в четвёртой четверти координатной плоскости.

Варинат 1

```
if x > 0 and y < 0:
  print("Да, четвёртая четверть")
else:
  print("Нет")
```

Варинат 2

```
if x > 0:
  if y < 0:
    print("Да, четвёртая четверть")
  else:
    print("Нет")
else:
  print("Нет")
```

Условный оператор. Примеры

Определить находится ли точка в четвёртой четверти координатной плоскости.

Варинат 1

```
if x > 0 and y < 0:
  print("Да, четвёртая четверть")
else:
  print("Нет")
```

Варинат 2

```
if x > 0:
  if y < 0:
    print("Да, четвёртая четверть")
  else:
    print("Нет")
else:
  print("Нет")
```

А здесь какой из вариантов лучше?

Оператор выбора

Что если нужно совершить то или иное действие в зависимости от значения одной переменной?

Например в зависимости от имени вывести на экран правильно обращение.

Александр -> Здравсвуйте, уважаемый Александр!
Александра -> Здравсвуйте, уважаемая Александра!
Саша -> Здравсвуй, Саша!

Оператор выбора

```
if name == "Александр":
    print("Здравсвуйте, уважаемый", name, "!")
else:
    if name == "Александра":
        print("Здравсвуйте, уважаемая", name, "!")
    else:
        if name == "Саша":
            print("Здравствуй, ", name, "!")
```

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в print

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в print

Но, оператора выбора в Python нет :(

Оператор выбора

Чтобы сократить код можно было бы использовать оператор выбора.

В котором каждому из вариантов поставить в соответствие подходящее обращение в print

Но, оператора выбора в Python нет :(

Зато есть оператор **elif** который можно использовать вместо `else + if`

Оператор elif

```
if   name == "Александр":
    print("Здравсвуйте, уважаемый", name, "!")
elif name == "Александра":
    print("Здравсвуйте, уважаемая", name, "!")
elif name == "Саша":
    print("Здравствуй, ", name, "!")
```

Все операторы на одном уровне вложенности и код читается легче. Особенно если расположить выражения в условии строго одно под другим.

Оператор elif

elif как и опаратор if можно дополнить оператором else.

```
if   name == "Александр":
    print("Здравсвуйте, уважаемый", name, "!")
elif name == "Александра":
    print("Здравсвуйте, уважаемая", name, "!")
elif name == "Саша":
    print("Здравствуй, ", name, "!")
else:
    # оператор else работает как обычно
    print("Здравствуй!")
```

Рекомендации

- Создавайте короткие и понятные условные операторы
- Проверяйте, все ли варианты учтены
- Используйте алгебру логики (алгебру высказываний) для упрощения логических выражений

Логические выражения

Логическое выражение — выражени результатом вычисления которой является «истина» или «ложь»

```
True
False
2 == 2
2 * 2 == 5
"war" == "peace"
2 * x == 5
not a
b or a and c
x > 10
z <= 0.5
sin(pi) > 0
```

Пустой оператор

pass – оператор который не выполняет никаких действий

Используется там, где синтаксис требует наличия оператора, но алгоритм не требует выполнения действий.

Также может использоватся как "заглушка когда нужно представить общий алгоритм, а его детали реализовать позднее.

Пустой оператор

```
import random from random
x = random()
if x > 0.5:
    pass  # todo: ...
else:
    pass  # todo: ...


# функция которая ничего не делает
# но в неё будет добавлен код в будущем
def my_function():
    pass
```

Ссылки и литература

- ru.wikibooks.org/wiki/Python - Викиучебник
- Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- Официальная документация Python3

  `help( имя )`

Приоритет операторов

1. `()`
2. `**` Возведение в степень
3. `-x, +x` унарные знаки + или минус.Например:`x=-x`
4. `*, /, //, %`
5. `+, -`
6. `==, !=, >, >=, <, <`
7. `not`
8. `and`
9. `or`

Приоритет операторов

```
x = 2**3 * ( 3 + 4) > 0  and  2 + 13//10 == 6

1.     8 * ( 3 + 4) > 0  and  2 + 13//10 == 6
2.           8 * 7  > 0  and  2 + 13//10 == 6
3.               56 > 0  and  2 + 13//10 == 6
4.               56 > 0  and  2 + 1 == 6
5.               56 > 0  and  3 == 6
6.                 True  and  False
6.                       False
```

Ссылки и литература

Материалы курса
github.com/VetrovSV/Programming